Classic Protocol

Message formats for the pedestrian

August 7, 1992

This note introduces the classic protocol message formats in use by the local control stations used at Fermilab in the D0 and Linac control systems. The messages can be used on token ring with SAP=$18, or they can be used from any IP node targetting UDP port# 6800 (decimal) for requests. For more extensive details, consult other documents.

Robert Goodwin
goodwin@fnal.gov

There are only a few basic classic protocol message type numbers as follows:

0. Data reply
1. (n.u.)
2. Data request
3. Data setting
4. Analog alarm
5. Digital alarm
6. Comment alarm

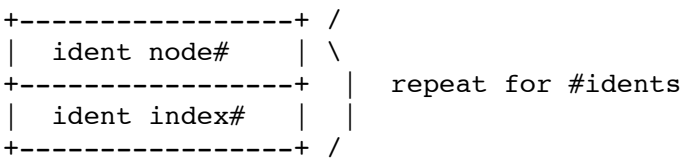
The following message formats are presented as arrays of 2-byte words in which the big endian byte order is assumed (hi byte first).
Hex data is used when possible in the format pictures. The message type number is found in the most significant 4 bits of the third word.

More than one message can be placed into a single UDP datagram.
The message size word in that case delineates each message.

The request message type is described before the reply.

Message type 2. Data request

```
+-----------------+
|  message size   |  message size (including this word) in bytes
+-----------------+
|  dest node#     |  node# to which this message sent.
+--------+--------+
|  2  s  | list#  |  $20= non-server. $28= server. list# identifies request.
+--------+--------+
| period | #ltyp  |  period in 15Hz cycles (0=one-shot), #listypes.
+----+---+--------+
| il |  #idents   |  ident length in bytes (4 bits), #idents (12 bits)
+----+---+--------+
| ltyp#  |  0  0  | \
+--------+--------+  |  repeat for #listypes
| #bytes req'd    |  |
```



The example shown is for the case of one listype, one ident, ident length=4. There are about 80 listypes defined. The meaning is roughly the kind of data that is desired. It implies a certain ident type and therefore ident length. Simple examples of listypes are:

listype#	kind of data requested	ident type
0	analog channel reading	channel#
1	analog channel setting	channel#
2	analog channel nominal value	channel#
3	analog channel tolerance value	channel#
4	analog channel alarm flags, count	channel#
20	memory data by bytes	address
21	digital bit I/O	bit#
25	digital byte I/O	byte#
29	memory data by words	address
40	analog reading in engineering units	channel#
41	analog setting in engineering units	channel#

The meaning of the request is that listype data is requested for all listypes using all idents given. For each listype and #bytes, generate response data for each ident. The ident array is processed in sequence for each listype. Thus a request is in general a matrix request. Therefore requests cannot be combined if one needs separate arrays of idents for each listype. In that case, more than one request should be made, each using a diffrent list#. If one wants 1 Hz readings and settings for channels 101,102,108 in node 576, then the request would be as follows:

```
001E  size=30
0576  dest node
2033  non-server request using list#=$33
0F02  1 Hz, 2 listypes
4003  4-byte idents, 3 idents
0000  reading
0002  2 bytes
0100  setting
0002  2 bytes
0576  target node/chan list
0101
0576
0102
0576
0108
```

In this non-server case, the dest node of the request message must match the target node# in each ident. In the server case, the request will be sent (as a non-server message) by the dest node to the target node, or in the case of more than one node in the list of idents that is not the same as the dest node, to a group (multicast) destination address to reach all local stations. In the group case, only stations which match at least one of the nodes in the list of idents will respond at all, including only their own contribution. The separate contributions are arranged in original request order and returned in a single reply. For this to work best the stations should run synchronized.

Message type 0. Data Reply

```
+-----------------+
|  message size   |  message size (including this word) in bytes
+-----------------+
|  dest node#     |  node# to which this message sent.
+--------+--------+
|  0  s  | list#  |  $00= non-server. $08= server. list# identifies request.
+--------+--------+
|   status word   |  reply status return code. 0=no error.
+-----------------+
|    reply data   |
+-----------------+
|        "        |
+-----------------+
```

This example shows the case of 4 data bytes returned. The dest node will be the node# of the requester node. For requesters that are not local stations there is no "node#", so this word will be zero. There is a single status word for the entire reply.

For the specific example given for the request message example, the following might be a reply:

```
0014  message size
0000  (zero dest node# for non-local station requester)
0033  reply type, non-server, list#=$33.
0000  status=0 (no errors)
1234  reading for chan 101
151D  reading for chan 102
4866  reading for chan 108
1230  setting for chan 101
1508  setting for chan 102
4882  setting for chan 108
```

Message type 3. Data setting

```
+-----------------+
|  message size   |  message size (including this word) in bytes
+-----------------+
|  dest node#     |  node# to which this message sent.
+--------+----+---+
|  3  s  | 0  |idw|  $30= non-server. $38= server. ident size in WORDS.
+--------+----+---+
| ltyp#  |  0  0  |  listype# for this setting
+--------+--------+
|     #bytes      |  #bytes of setting data
+-----------------+
|  ident node#    | \
+-----------------+  |  setting ident
|  ident index#   |  |
+-----------------+ /
|  setting data   |
+-----------------+
```

One setting message targets one device by listype# and ident. An example of a setting to channel $300 in station 576 would be as follows:

```
0010  size
0576  dest node
3002  setting, 2 word ident
0100  setting listype#
0002  2 bytes of data
0576  target node#
0300  channel 0300
1234  setting data value
```

In this non-server case, the dest node of the setting message must match the target node# in the ident. In the server case, the setting will be sent (as a non-server message) by the dest node to the target node.

Message type 4. Analog alarm

```
+-----------------+
|  message size   |  message size (including this word) in bytes
+-----------------+
|  dest node#     |  node# to which this message sent.
+--------+--------+
|  4  0  |  0  0  |  analog alarm
+--------+--------+
|    channel#     |  analog channel# in alarm
+-----------------+
|   alarm flags   |  flag bits. bit#8 (mask $0100) is good/bad bit. 1=bad.
+-----------------+
```

```
|    reading      |  analog reading
+-----------------+
|    setting      |  analog setting
+-----------------+
|    nominal      |  nominal value
+-----------------+
|    tolerance    |  tolerance value
+-----------------+
|    spare        |  spare word
+--------+--------+
|   'A'  |   'B'  |  analog 6-char channel name (example name='ABCDEF')
+--------+--------+
|   'C'  |   'D'  |
+-----------------+
|   'E'  |   'F'  |
+-----------------+
|  year  | month  |  time of alarm detected in BCD
+-----------------+
|  day   |  hour  |
+-----------------+
| minute | second |
+-----------------+
| cycle  |  0  0  |
+-----------------+
|                 |
+----fullscale----+  reading fullscale conversion constant (IEEE flt pt)
|                 |
+--------+--------+
|                 |
+-----offset------+  reading offset conversion constant  (IEEE flt pt)
|                 |
+--------+--------+
|   'V'  |   'O'  |  engineering units text (example='VOLT')
+-----------------+
|   'L'  |   'T'  |
+-----------------+
```

An analog alarm message is sent for any good/bad transition of an analog channel that is in the alarm scan (bit#15 of alarm flags word set). Examine bit#7 of the alarm flags word for a good(0) or bad(1) message. The analog alarm scan has a hysteresis to prevent chatter. Once a channel's reading has drifted outside the tolerance window from the nominal value, and a "bad" alarm message is generated, the reading must be found within half a tolerance before a "good" message is generated.

Inclusion of the scaling constants permits calculation of the reading in engineering units at the time the alarm (good or bad) message was generated. The engineering units scaling is linear and uses the formula:

units = float(raw)*fullscale/32768. + offset

A specific example of an analog alarm message is:

```
002E  size
00F0  group dest node
4000  analog alarm
0150  channel 0150
B000  active(8000), inhibit(2000), two-times(1000), good(0100)
C596  reading
0000  setting (motor-controlled device has no setting value)
C5BC  nominal
0064  tolerance
0000  spare
4848  HHVOLT (H- preaccelerator high voltage)
564F
4C54
9208  time of alarm: 08/05/92 1648:03, cycle 14.
0516
4803
1400
44CC  fullscale  1638.
C000
0000  offset     0.
0000
4B56  units text 'KV  '
2020
```

The computed engineering units reading value in this case is -747.5 KV.

Message type 5. Digital alarm

```
+-----------------+
|  message size   |  message size (including this word) in bytes
+-----------------+
|  dest node#     |  node# to which this message sent.
+--------+--------+
|  5  0  |  0  0  |  digital alarm
+--------+--------+
|      bit#       |  digital bit# in alarm
+-----------------+
|   alarm flags   |  flag bits. bit#8 (mask $0100) is good/bad bit. 1=bad.
+-----------------+
|   'P'  |   'R'  |  digital 16-char bit text  (example='PREACC AIR COND ')
+--------+--------+
|   'E'  |   'A'  |
+-----------------+
|   'C'  |   'C'  |
+-----------------+
|   ' '  |   'A'  |
+--------+--------+
```

```
|   'I'  |   'R'  |
+-----------------+
|   ' '  |   'C'  |
+--------+--------+
|   'O'  |   'N'  |
+-----------------+
|   'D'  |   ' '  |
+-----------------+
|  year  | month  |  time of alarm detected in BCD
+-----------------+
|  day   |  hour  |
+-----------------+
| minute | second |
+-----------------+
| cycle  |  0  0  |
+-----------------+
```

A specific example of a digital alarm might be as follows:

```
      0022  size
      00F0  group dest node
      5000  digital alarm
      019C  bit 019C
      C100  active(8000), nominal=1(4000), bad(0100)
      4354  bit text (16 chars) 'CTF NTFREQ      '
      4620
      4E54
      4652
      4551
      2020
      2020
      2020
      9208  time of alarm: 08/07/92 1115:22, cycle 1.
      0711
      1522
      0100
```

Message type 6. Comment alarm

```
+-----------------+
|  message size   |  message size (including this word) in bytes
+-----------------+
|  dest node#     |  node# to which this message sent.
+--------+--------+
|  6  0  |  0  0  |  comment alarm
+--------+--------+
|    comment#     |  comment index# of alarm
+-----------------+
|   alarm flags   |  flag bits. (no good/bad bit for comments)
+-----------------+
|   'V'  |   'M'  |  comment 16-char text  (example='VME ALARM RESET ')
+--------+--------+
```

```
|   'E'  |   ' '  |
+----------------+
|   'A'  |   'L'  |
+----------------+
|   'A'  |   'R'  |
+--------+--------+
|   'M'  |   ' '  |
+----------------+
|   'R'  |   'E'  |
+--------+--------+
|   'S'  |   'E'  |
+----------------+
|   'T'  |   ' '  |
+----------------+
|  year  | month  |  time of alarm detected in BCD
+----------------+
|  day   |  hour  |
+----------------+
| minute | second |
+----------------+
| cycle  |  0  0  |
+----------------+
```

There are only two comment alarms in current use. One is the system reset message, and the other is the alarms reset message. A specific example of a comment alarm might be as follows:

```
0022  size
00F0  group dest node
6000  digital alarm
0000  comment index 0
8000  active(8000)
564D  comment text (16 chars) 'VME SYSTEM RESET'
4520
5359
5354
454D
2052
4553
4554
9208  time of alarm: 08/07/92 1129:20, cycle 5.
0711
2920
0500
```